Advanced Algorithms

Date:  Wednesday 2nd June 2010
Time: 09.45 – 11.45

Please answer any THREE Questions from the FIVE questions provided
Each question carries equal marks
This is a CLOSED book examination

The use of electronic calculators is NOT permitted.
1. a) Describe the Boyer-Moore string searching algorithm which finds the first occurrence of a pattern in a text by scanning the pattern right-to-left, but moving the pattern left-to-right. Your description should:

i) explain clearly the two pre-processing procedures applied to the pattern. (6 marks)

ii) explain clearly the matching phase of the algorithm. (4 marks)

You need not give a program (or a pseudocode description) but should give a clear description of the steps of the algorithm.

b) Consider the following technique for searching a text string for the first occurrence of a pattern: for each character of the alphabet, a list is formed of all positions of that character in the pattern (the list is empty if the character does not occur in the pattern). Then iterate through the text examining only every \( m \)-th character, where \( m \) is the length of the pattern. For each such character in the text, we examine each position that this character occurs in the pattern, and attempt to match the pattern character-by-character at that position.

i) Give a full description of this algorithm. Your description may be a program or a pseudocode description or a precise step-by-step explanation. (6 marks)

ii) Give an analysis of the time and space complexity of the algorithm, including worst-case, average-case and best-case time complexities. (4 marks)
2.  
   a) Explain what is meant by a depth-first search (DFS) of a directed graph. (2 marks)

   b) Give (as a program or in pseudocode) a recursive algorithm for the DFS of directed graphs which allocates a number to each node of a graph according to the order in which nodes are visited. (4 marks)

   c) What is the time complexity of your DFS algorithm in terms of the number of nodes N and number of edges E of a graph? (2 marks)

   d) Explain clearly how to modify your DFS algorithm to keep track of ancestry in the collection of DFS trees, i.e. for each edge encountered, we determine whether or not it links a node with an ancestor in a DFS tree. You need not express this modification as a program or in pseudocode, but you should explain clearly what data structures you introduce and how they are used to record ancestry. Your solution should still result in a linear-time DFS algorithm. (4 marks)

   e) Explain how this modified algorithm may be used to classify all edges of a graph with a DFS into tree edges, back edges, forward edges or cross edges. (2 marks)

   f) How many trees are in a DFS forest of a graph which contains just one strongly connected component? Explain your answer. (2 marks)

   g) Consider now your recursive algorithm for the DFS numbering of nodes. Suppose the starting node and any re-start nodes are chosen at random, so that a re-run of the algorithm may choose different starting nodes. Consider a series of such runs and, for each run, we count the number of trees in the DFS forest. This series of numbers provides some probabilistic information about the graph. What information can be extracted from such a series of runs? [Hint: Consider the strongly connected components of a graph and the connections between these.] (4 marks)
3. a) It is widely stated that “model checking is probably the most successful automated analysis technology to have been delivered by formal methods research”. Briefly explain the underlying concepts of model checking and then give justifications for the above quotation. (4 marks)

b) State briefly what a Kripke structure is and then explain how, in general terms, such a structure can be used to represent the execution behaviour of finite state concurrent programs. (6 marks)

c) Build a Kripke structure that represents the execution behaviour of the following concurrent program.

```plaintext
1:   while (true) do
2:   cobegin
3:     L1: x = (y+1) mod 3;
4:     L2:
5:   ||
6:     R1: y = (x+1) mod 2;
7:     R2:
8:   coend
```

in which the assignment statements at lines 3 and 6 are executed concurrently, but atomically, within the body of the while loop. You may find it helpful to assume a program counter `pc` that takes values according to the following table

<table>
<thead>
<tr>
<th>Label Pairs</th>
<th>pc Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1, R1</td>
<td>1</td>
</tr>
<tr>
<td>L2, R1</td>
<td>2</td>
</tr>
<tr>
<td>L1, R2</td>
<td>3</td>
</tr>
<tr>
<td>L2, R2</td>
<td>1</td>
</tr>
</tbody>
</table>

You should assume that the variables `x` and `y` are restricted to finite integers, in the range 0 to 2, and have initial values of 0. (6 marks)

d) Outline an algorithm for computing the forwards reachable set of states, from the given initial state, of a given Kripke structure. Apply your algorithm to the Kripke structure obtained in answer to part (iii) above to show that a state that has `x==1` and `y==2` can never be reached. (4 marks)
4.  

a) What are strongly connected components (SCCs) of a graph, and what are terminal SCCs? Exemplify your answer by giving a graph containing just two nodes and containing two different SCCs, one of which is also terminal. (4 marks)

b) Explain the relevance of finding SCCs (and terminal ones) to the problem of determining whether a given infinite word automaton (with Buchi acceptance conditions) represents a non-empty language. (4 marks)

c) Outline an algorithm for finding the SCCs of a graph that contain an accepting state and give a brief justification for its correctness. (4 marks)

d) Use the algorithm you provide in answer to part (iii) above to determine whether the intersection of the languages represented by the following two \( \omega \)-regular expressions

i) \( a(bc) \omega \)

ii) \( a b^* (b|c) \omega \)

is empty. (8 marks)
5. a) Explain what is meant by a time-bounded polynomial algorithm and a space-bounded polynomial algorithm. Give one example of a linear space-bounded algorithm and one example of a quadratic time-bounded algorithm. (2 marks)

b) What is meant by a polynomial-time reduction of one computational problem to another? (2 marks)

c) What is meant by a computational problem being NP-complete? (2 marks)

d) Explain clearly how a polynomial-time reduction may be used to establish the NP-completeness of a computational problem, if we are given another problem known to be NP-complete. (4 marks)

e) In an undirected graph, an independent set of nodes, is a set of nodes no two of which are linked by an edge. Consider the problem of determining whether or not a graph has an independent set of nodes of size k.

i) Show that this problem is in NP.

ii) By defining a reduction from the problem of determining whether a propositional formula is satisfiable, show that the above problem of finding independent sets is NP-complete. [Hint: consider the relationship of the problem of finding independent sets to that of finding cliques in a graph and use this to define a reduction.] (10 marks)